Thesis

IQ-Mower: PN-sequence-transmission using Bluetooth.

Ann-Britt Carlsson
Supervisor: Peder Carlsson
2002-06-24

Högskolan Trollhättan-Uddevalla
Institutionen för Teknik
Box 957, 461 29 Trollhättan
Tel: 0520-47 50 00 Fax: 0520-47 50 99

Foreword:

It has been interesting to study the Bluetooth technology, I would like to Thank my supervisor Peder Carlsson for letting me have this study as my examination.

I would also like to thank Mattias Ottosson for helping me come into the Visual C++ programming and for his wonderfully smile when I managed to transmit the first A, and thanks to Gunne Andersson for making a cable connection specially for my Blue-tooth experiments.

Thanks also to Lisbeth for answering my strange questions and wonders in the beginning of this work. You were right, I do laugh towards them today.

And special Thanks to Pelle and the rest of my family for letting me study all those evenings and weekends.

Abstract:

The IQ-mower is a new type of lawn mower. It will cut the grass all by itself in an “intelligent” way. Three beacons are going to send noise sequences to it. If the IQ-mower knows how long time the noise sequences took from the beacons to the IQ-mower it will calculate were on the lawn it is. The beacons need to send the sequence once every second but not all in the same time. The IQ-mower has to tell which beacon to transmit to and when to transmit. It will only get the noise sequence in return, any re-transmission of the “turn on” sequence is not allowed. If any transmission fails the IQ-mower has to wait for the next one.

Bluetooth is a new communication technology. It can work in a piconet with one master and up to seven slaves. This project has researched if Bluetooth can be used in the IQ-mower system to send the turn on sequences from the IQ-mower to the beacons.

The Ericsson Bluetooth Application & Training Tool Kit has been used in this project. The Test Sample Application in the Kit has been modified to test if it will fit the IQ-Mower. Due to the fact that the two Bluetooth modules in the kit were only point-to-point, it has not been possible to try the application with three beacons.

If hardware for an embedded solution, with full stack on chip and application on host, is used and SCO or AUX1 Bluetooth packets transmit the “turn on” sequences it will be possible to use Bluetooth in the IQ-mower system.

Index:

1 INTRODUCTION 8

2 AIM OF STUDY 8

3 METHOD/MATERIAL 8

4 WHAT IS BLUETOOTH? 9

4.1 How it started. 9

4.2 Bluetooth Special Interest Group (SIG). 9

4.3 Royalty-free license 10

4.4 Worldwide operation 10

4.5 Frequency Hopping 11

4.6 Slots 12

4.7 Piconet and Scatternet 12
4.7.1 Point-to-point and point-to-multipoint 13

4.8 Master and Slave 14

4.9 Physical Links 14
4.9.1 ACL Link 14
4.9.1.1 AUX1 15
4.9.2 SCO Link 15

4.10 Packet Definition 16
4.10.1 Access Code 16
4.10.2 Header Code 16
4.10.3 Payload 17
4.10.4 Error correction 17
4.10.5 Multi-slot packets 18

4.11 Bluetooth Clocks 18

4.12 Unique Bluetooth Address 19

4.13 Security 19

4.14 Health 19

5 THE BLUETOOTH ARCHITECTURE 20

5.1 The Bluetooth Device 20
5.1.1 The Host and the Bluetooth Module 21

5.1.2 The Module used at HTU 21
5.1.3 Three Different Bluetooth Implementation Models. 21

5.2 The Lower Layers of the Stack 23
5.2.1 The Bluetooth Radio 24
5.2.2 Bluetooth Baseband 24
5.2.3 The Link Manager (LM) 25
5.2.4 The Host Controller (HC) 25
5.2.4.1 Bluetooth Traffic Light Utility (BTU) 25

5.3 The Bluetooth Profiles 26
5.3.1 New Profiles 27

6 THE ERICSSON SYSTEM LEVEL DESCRIPTION 28

6.1 The Upper Stack 28
6.1.1 HCI (The Host Controller Interface) 29
6.1.2 L2CAP (Logical Link control and Adaptation Protocol). 29
6.1.3 RFCOMM 30
6.1.4 SDP (Service Discovery Protocol) 30
6.1.5 OBEX 30
6.1.6 TCS (Telephony Control protocol Specification) 31

6.2 COM-server 31

6.3 HCI packets 32
6.3.1 HCI Command packet 32
6.3.2 HCI Event packets 33
6.3.3 HCI Data packet 33

6.4 API (Application Programmers Interface) 34
6.4.1 API supported profiles 34
6.4.2 The stack API 34
6.4.3 Global API Description 35
6.4.4 Layer to layer 36
6.4.5 Start 38
6.4.6 Registration & DeRegistration 38
6.4.7 Connection & DisConnection Set up 39
6.4.8 Data transmission 40
6.4.9 The Page and Inquiry Procedures 41
6.4.9.1 The Inquiry Procedure 41
6.4.9.1.1 Inquiry Result Filter 42
6.4.9.2 The Page Procedure 43
6.4.9.3 Avoiding collisions 43

7 TEST SAMPLE APPLICATION SUITE 43

7.1 Test Sample application 44

7.2 My Modified Test Sample 45

8 BLUETOOTH AND FPGAS 46

9 RESULTS 46

10 DISCUSION 46

11 CONCLUSIONS 47

12 REFERENCES 48

13 APPENDIX 48

Figure index:

Figure 1.1 The IQ-Mover radio system..... 8
Figure 4.1 Slot timing for single-slot packets..... 12
Figure 4.2 Schematic figure over piconet and scatternet..... 12
Figure 4.3 Point-to-point and point-to-multipoint piconets..... 13
Figure 4.4 An example of different links to different Slaves. 16
Figure 4.5 Standard packet format..... 16
Figure 4.6 Bit-repetition encoding scheme. 17
Figure 4.7 One time slot, three time slots and five time slots..... 18
Figure 5.1 Bluetooth Application Tool Kit. 21
Figure 5.2 The three stack implementation models. 22
Figure 5.3 The different layers of the Bluetooth Module..... 23
Figure 5.4 BTU main window..... 26
Figure 5.5 Bluetooth profiles implemented by Bluetooth version 1.0. 27
Figure 6.1 An overview of the PC Reference Stack and its components. 28
Figure 6.2 The Com-server and Com-client on top of the Module..... 31
Figure 6.3 HCI command packets. 33
Figure 6.4 HCI Event packet..... 33
Figure 6.5 HCI ACL Data packet. 33
Figure 6.6 HCI SCO Data Packet. 34
Figure 6.7 Inter-process communication within the Bluetooth Host Stack. 35
Figure 6.8 Communication principles between different layers. 35
Figure 6.9 Command & Event..... 36
Figure 6.10 The start procedure..... 38
Figure 6.11 Registration and DeRegistration. 39
Figure 6.12 Connection and disconnection sequence. 40
Figure 6.13 Data transmission..... 41
Figure 6.14 Inquiry. 42
Figure 7.1 How the different parts of the sample application interacts with each other, the stack and VOS. 44
Figure 7.2 My modified Test Sample Client. 45
Figure 7.3 My modified Test Sample Server..... 46

Table index:

Table 4.1 Regional Bluetooth frequency limitations. 11
Table 5.1 Output Power for Each Bluetooth Power Class.:..... 24
Table 6.1 Communication from layer A to layer B. 36
Table 6.2 Communication from layer B to layer A. 37
Table 7.1 File description of the Test Sample Application. 44

1 Introduction

At HTU there is a project going on with an intelligent lawn mower. This lawn mower is going to cut the grass all by itself. It is named the IQ-Mower. In Figure 1.1 you can se the radio-system that is going to help the IQ-Mover to know were it is. The micro-processor, FPGA, A/D, AGC and RS will be placed in the IQ-Mower, the RM, CPLD and power amplifier will be placed in a beacon. There will be at least three beacons in one system.

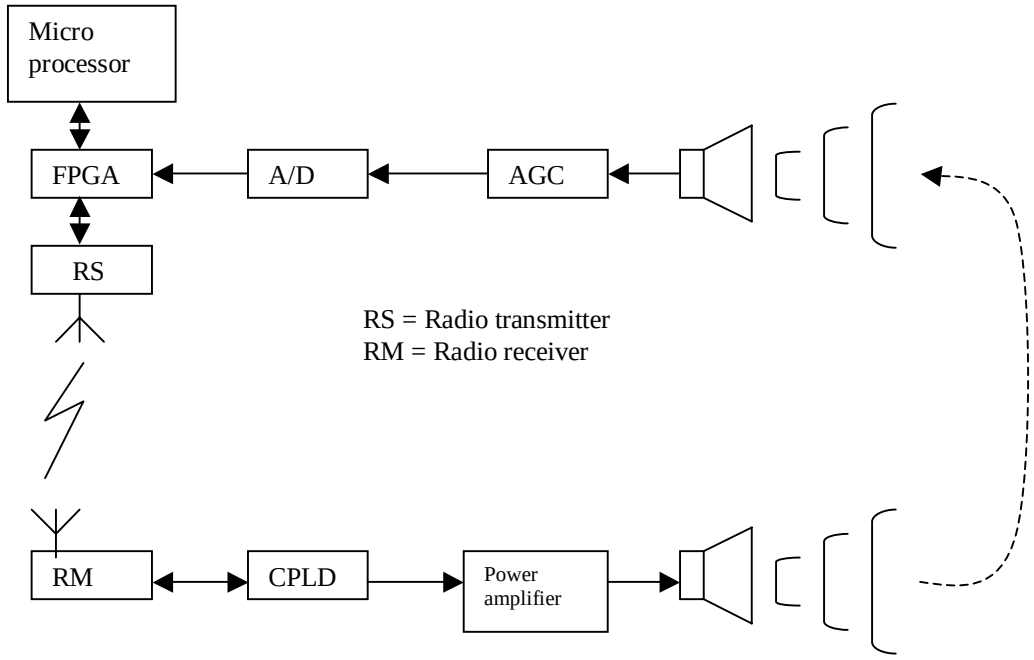


Figure 1.1 The IQ-Mover radio system.

The IQ-Mover needs to send a sequence to every beacon, one by one, once every second to start the CPLD. We do not want any retransmission of the sequence. If any transmission fails the beacon have to wait for the next transmission.

2 Aim of study

The aim of this study is to investigate whether it is possible to use Bluetooth in the radio transmitter and receiver above.

3 Method/Material

In this degree thesis the following products have been used.

- Ericssons Bluetooth Application & Training Tool Kit.
- Ericssons Bluetooth PC Reference Stack R1a, and R2B.
- Bluetooth Specification, version 1.0 and 1.1.
- Visual C++.
- Word 2000.
- Windows 98.

4 What is Bluetooth?

It is a new type of smart wireless communication. The Bluetooth system is a universal radio interface on the globally available 2.4 GHz frequency band facilitating wireless communication of data and voice in both stationary and mobile environments. Bluetooth is a technology that promises to eliminate most of the cables that connect various personal computing-devices, and to create new types of smart wireless communications. Bluetooth technology will enable all different electronic devices to connect to each other – without wires.

Bluetooth is more than just a cable-replacement technology. It is also a technology that enables any electronic device to communicate with any other electronic device, automatically. This means, over short distances (up to 100 meter), your cell phone or personal digital assistant (PDA) can connect to, synchronise with, and even control the other electronic devices in your home or office – such as your personal computer, printer, television set, home alarm system, or home/office telephone system. All of this communication can take place without your being aware, totally automatically.

4.1 How it started.

The idea that eventually became Bluetooth was born in 1994, when engineers at Ericsson Mobile Communications investigated the feasibility of a low-power, low-cost radio interface between mobile phones and their accessories.

4.2 Bluetooth Special Interest Group (SIG).

In 1997 Ericsson approached several manufacturers of portable electronic devices to discuss the development of this new short-

range wireless technology. And in 1998 five key companies, Ericsson, IBM, Intel, Nokia and Toshiba formed the Bluetooth Special Interest Group (SIG), to coordinate the development of and promote the Bluetooth technology. Bluetooth was formally announced in May of 1998, and the Bluetooth SIG released version 1.0 of the Bluetooth specification in July of 1999. [1]

Since then membership in the Bluetooth SIG has grown to include more than 2400 different companies. Four other large companies 3Com, Lucent Technologies, Microsoft and Motorola have joined the five founding companies as so called promoter members. These nine companies provide direction and promotion for the entire 2000-company Bluetooth effort. You can read more about SIG on www.bluetooth.com .

Bluetooth has been adopted by all the major players in the telecom and computer worlds, as well as an interesting cross-section of companies in other industries including the home entertainment, automotive, health care, industrial automation, and toy industries.

4.3 Royalty-free license

The Bluetooth technology itself is available to all member companies via a royalty-free license. Just about any company can use Bluetooth technology at no charge.

All new Bluetooth radios and profiles have to be tested by SIG to manage the standard.

4.4 Worldwide operation

One of the key requirements in the design of the air-interface was that it should be able to operate worldwide. The only frequency band that satisfies these requirements is at 2.4 GHz - the Industrial-Scientific-Medical (ISM) band. The ISM band is also license free and open to any radio system.

Since the ISM band is open to anyone, radio systems operating in this band must cope with several unpredictable sources of interference, such as:

- baby monitors
- cordless phones
- garage door openers
- microwave ovens

Microwave ovens are the strongest sources of interference, but in most cases you only need to move the Bluetooth device a little bit (one or two meters) away from the oven to get less interference [2].

Three countries Spain, France and Japan had previously reserved portions of the 2.4GHz range for local (primarily military) use. This could have potentially defeated the global nature of the Bluetooth standard if devices built for these three countries had to incorporate slightly different radio technology. The Bluetooth SIG has worked with officials from each country to try to lift restrictions on 2.4GHz transmissions. As of January 2001 all three countries have agreed (at least in principle) to free up at least some of the reserved portions of the band in their respective regions.

Table 4.1 compares the frequency ranges available for Bluetooth use in the problematic regions with the standard 2.4GHz band available elsewhere in the world.

Table 4.1 Regional Bluetooth frequency limitations.

Country	Frequency Range (GHz)	Available Channels
Japan	2.471 – 2.497	23
France	2.4465 – 2.4835	23
Spain	2.445 – 2.475	23
Elsewhere (Europe and U.S)	2.400 – 2.483	79

There are functions to set different country codes.

4.5 Frequency Hopping

The operating band of 83.5 MHz is divided into 1 MHz spaced channels each signalling data at 1M Symbols per second so as to obtain the maximum available channel bandwidth. This split up gives 79 available channels. (Japan, France and Spain: 23 channels). A data channel hops randomly 1,600 times per second between the 79 (or 23) channels.

4.6 Slots

Each channel is divided into 625 µs time slots. A TDD (Time-Division Duplex) scheme is used which means that the time slots are used alternately for transmitting and receiving. One packet can be transmitted per time slot. The master transmits a packet on its time slot k on frequency f(k) to a slave. In the next time slot k+1 on frequency f(k+1) it receives a packet from the slave before it transmits again. Note the packet does not occupy the full time slot (625 µs) as time is reserved to allow the device to switch between transmission and receiving modes. Figure 4.1 describes this.

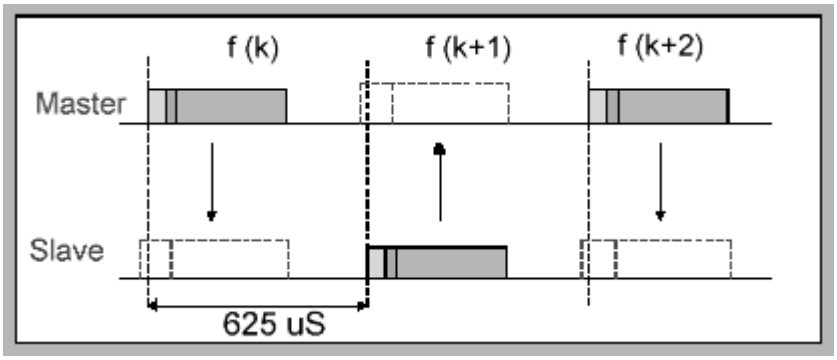


Figure 4.1 Slot timing for single-slot packets.

4.7 Piconet and Scatternet

The frequency-hopping scheme also enables Bluetooth to work in not only point-to-point connections but also in point to multipoint connections. This means that two or more units sharing the same channel form a piconet, consisting of one master and up to seven slaves. When more than one master is involved in the piconet then they form a scatternet. Figure 4.2 shows the two concepts.

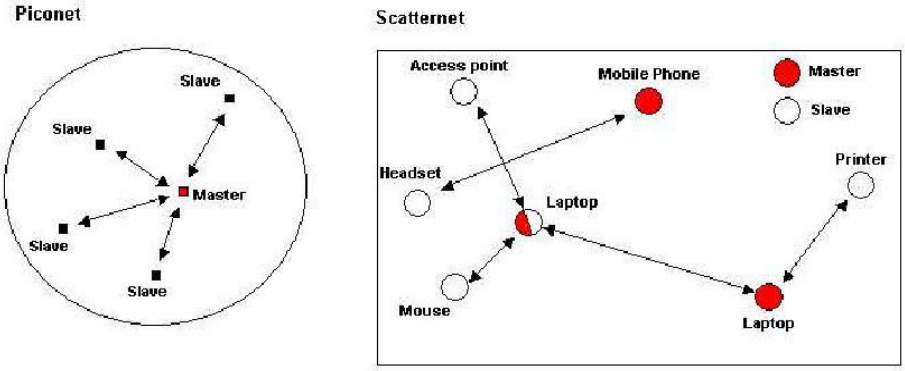


Figure 4.2 Schematic figure over piconet and scatternet.

The Bluetooth devices have to be in range of each other to establish connections they also have to be of type point-to-multipoint (see chapter 4.7.1) to be able to establish piconet or scatternet connections.

Once multiple links are in place, the device must handle QOS (Quality of Service) balancing between the links. That is to say, it must decide how much bandwidth to allocate to each link.

The devices in a piconet share a common communication data channel. The channel has a total capacity of 1 megabit per second (Mbps). Headers and handshaking information consume about 20 percent of this capacity.

If two piconets are close by, each set of Slaves will only receive packets on their own piconet, which have that piconets sync word. Similarly a Master will only receive from its own Slaves.

As mobile devices move around, they can easily move out of range and lose contact with the piconet. Each link has a supervision timeout, such ensures that such links are closed down.

4.7.1 Point-to-point and point-to-multipoint

With a point-to-point Bluetooth module you can only establish one link to one other Bluetooth device. With a point-to-multipoint module you can establish several links between one master device and up to seven slaves, ACL connections (only three with SCO connections)

Figure 4.3 Point-to-point and point-to-multipoint piconets.

The type of Ericssons Bluetooth modules you can identify from the sticker on the hardware. P/N 51-1106-xx is the point-to-point version, and P/N 51-1208-xx is the point-to-multipoint version.

Datasheets of the Ericsson Bluetooth Modules with the number rok101007 is point-to-multipoint and rok101008 is point-to-singlepoint functionality.

4.8 Master and Slave

Since the radios in all Bluetooth devices are essentially identical in function, any device can be a Master, and any can be a Slave. It is really a function of which device broadcasts first.

- Master - The device that initiates an exchange of data.
- Slave - The device that responds to the Master.

A piconet has one Master and up to seven Slaves. The Master transmits in even time slots, Slaves in odd time slots.

4.9 Physical Links

A link is established between a device acting as Master and a device acting as Slave. Once a link has been set up, there are two basic types of data packets that may be exchanged with one very important distinction between the two; each is characterised by the two following link types:

- Asynchronous Connection-Less (ACL).
- Synchronous Connection Oriented (SCO).

4.9.1 ACL Link

An ACL link exists between a Master and a Slave as soon as a connection has been established. A Master may have a number of ACL links to a number of different Slaves at any one time, but only one link can exist between any two devices.

ACL links support symmetrical or asymmetrical, packet-switched, point-to-multipoint connections and are typically used for data transmission.

ACL is either point-to-point (master to one slave) or broadcast to all the slaves. Broadcast packets are ACL packets that are not addressed to a specific Slave and so are received by every Slave.

ACL slaves can only transmit when requested by the master. A slave is allowed to return an ACL packet in the slave-to-master slot if and only if it has been addressed in the preceding master-to-slave slot.

Most ACL packets facilitate error checking and retransmission to assure data integrity.

4.9.1.1 AUX1

All of the ACL packets contain a 2-byte CRC (Cyclic Redundancy Checksum) code except for the AUX1 packet. AUX1 packets are DH1 (Data High rate) packets with the CRC replaced by data. Unlike all the other ACL packets, AUX1 packets are not retransmitted.

The AUX1 is for the moment, the only ACL packet that is not retransmitted.

4.9.2 SCO Link

SCO links support symmetrical, circuit-switched, point-to-point connections and are typically used for voice transmission. An example of a SCO link is the voice link between a cellular phone and headset. SCO links reserve two consecutive slots - one for transmission and one for reception - at fixed intervals.

A master can support up to three SCO links to the same slave or to different slaves. A slave can support up to three SCO links from the same master in a piconet or two SCO links if the links originate from different masters.

SCO packets are never retransmitted.

In a piconet, there can be up to three SCO links of 64 kbits per second each. To avoid timing and collision problems, the SCO links use reserved slots set up by the Master. Notice the SCO link, in figure 4.4, it sends every 6:th slot. A slave is allowed to send a SCO packet even if no packet was received in the previous master-slot.

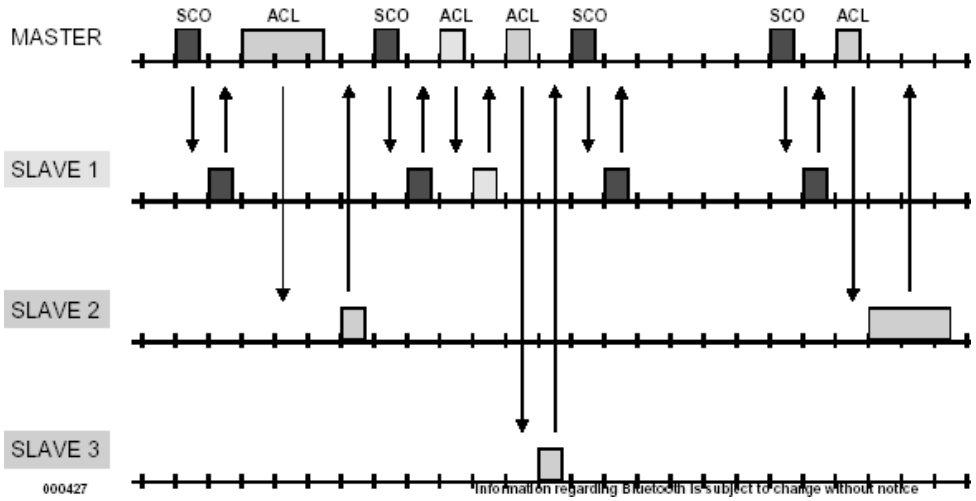


Figure 4.4 An example of different links to different Slaves.

Figure 4.4 shows how a Master is able to communicate with several Slaves in sequence. The Master always transmits on even slots and the Slave replies on odd-numbered slots.

In the slots not reserved for SCO links, the master can exchange packets with any slave on a per slot basis.

4.10 Packet Definition

The data on a channel is transmitted in packets. Data in a packet can be up to 2,745 bits in length. The standard packet format is shown in figure 4.5.

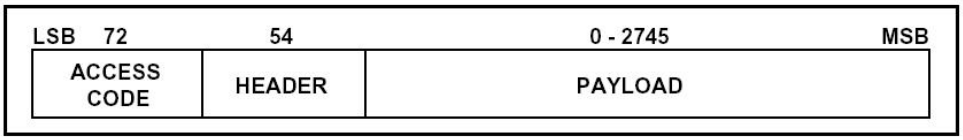


Figure 4.5 Standard packet format.

4.10.1 Access Code

Every packet exchanged on the physical channel has a 68/72-bit access code, derived from the BD_ADDR of the master and is unique for that channel. The access code identifies the piconet to whom the packet is going to be sent.

4.10.2 Header Code

The header code identifies the slave to who the packet is going to be sent.

4.10.3 Payload

The data you want to transmit is called the payload.

You can read more about packet format in Specification of the Bluetooth System [3] or in User Manual Bluetooth Pc Reference Stack by Ericsson version R1c [4].

4.10.4 Error correction

There are three error correction schemes defined for Bluetooth packets:

• 1/3 rate FEC (Forward Error Correction)

• 2/3 rates FEC

• ARQ (Automatic Repeat request) scheme for the data

The purpose of the FEC scheme on the data payload is to reduce the number of retransmissions. However, in a reasonable error-free environment, FEC gives unnecessary overhead that reduces the throughput. Therefore, different packet types have been defined.

FEC in the payload or not, resulting in the DM (Data Medium rate) and DH (Data High rate) packets for the ACL link and the HV (High quality Voice) packets for the SCO link, this packet can send data as well as voice.

The packet header is always protected by a 1/3 rate FEC; it contains valuable link information and should be able to sustain more bit errors.

A simple 3-times repetition FEC code is used for the header. The repetition code is implemented by repeating the bit three times, see the illustration in figure 4.6.

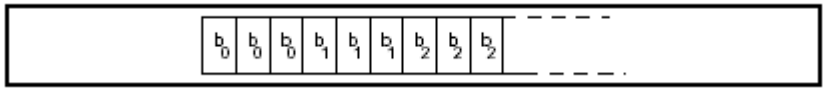


Figure 4.6 Bit-repetition encoding scheme.

The 3-bit repetition code is used for the entire header, and also for the voice field in the HV1 packet.

To determine whether the payload is correct or not, a cyclic redundancy check (CRC) code is added to the packet. The ARQ scheme only works on the payload in the packet (only that pay-

load which has a CRC). The packet header and the voice payload are not protected by the ARQ scheme.

4.10.5 Multi-slot packets

To support high data rates, multi-slot packets have been defined with a packet occupying one time slot, three time slots, or five time slots. However, the hop frequency used in the first slot will be used throughout the transmission.

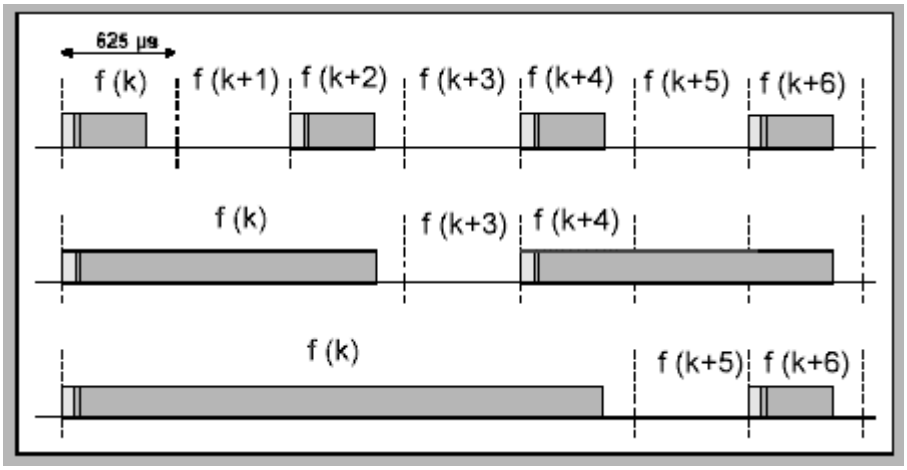


Figure 4.7 One time slot, three time slots and five time slots.

Figure 4.7 shows the three different sizes of packets. Notice the empty slots they are reserved for Slave to Master packets.

4.11 Bluetooth Clocks

Every Bluetooth device has an internal system clock, known as a native clock (CLKN), which determines the transceiver timing for that device. CLKN is never adjusted or turned off.

Piconet synchronisation is determined by the master's clock of the piconet (CLK). CLK is identical to the master's native clock, CLKN. Slaves in the piconet add a timing offset to their own native clock to synchronise with the master clock. To maintain piconet synchronisation, the timing offsets of the slave clocks are updated regularly - each time a packet is received from the master.

The master can request this clock offset anytime during a connection. By saving this clock offset the master knows on what RF channel the slave wakes up to PAGE SCAN (see chapter 6.4.9), after it has left the piconet. This can be used to speed up the paging time the next time the same device is paged.

4.12 Unique Bluetooth Address

Every Bluetooth device has a unique Bluetooth device address. The baseband part of the Bluetooth specification describes an algorithm that can calculate a frequency hop sequence from a Bluetooth device address and a Bluetooth clock. When Slaves connect to a Master, they are told the Bluetooth device address and clock of the Master. They then use this to calculate the frequency hop sequence. Because all Slaves use the Master´s clock and address, all are synchronised to the Master´s frequency hop sequence.

4.13 Security

The high-speed pseudo-random frequency-hopping algorithm employed in Bluetooth makes it very difficult to listen in on a Bluetooth connection. In fact, the U.S military considers a communications link using frequency hopping over 79 channels to be secure in it self [2].

Authentication prevents spoofing and unwanted access to critical data. For example, connecting a laptop to a cellular phone may require a PIN to be entered. Encryption prevents eavesdropping and maintains link privacy.

4.14 Health

Bluetooth uses frequency spectrum in the range of 2400MHz to 2483,5MHz. This range encompasses the natural frequency of H_2O molecular oscillation at 2450MHz, which is also used by microwave ovens specifically to excite water molecules inside food in order to cook it.

Sharing the same frequency range as microwave ovens has led to some concerns that Bluetooth devices might cook their users. Some microwave radiation will be absorbed in flesh. It will be absorbed to heat through molecular friction. Basically the microwaves shake the water in flesh and it heats up as it shakes. But as the radiated output power of Bluetooth devices is incredibly low and spread in spectrum in time experts concur that Bluetooth radiation does not pose a risk to health.

A 1mW Bluetooth radio emits 1/1 000 000 the amount of power in a 1KW microwave oven. Also in a microwave oven all the power is directed inward at the food while in a Bluetooth device the power is radiated outward so the user only ever intercepts the smallest fraction of the radio waves that are heading in their direction.

In comparing with other popular communications devices Bluetooth operates at 2.4GHz and uses 1mW (0dBm) for most applications with a maximum of 100mW (20dBM) for extended range. This means that Bluetooth signals have a penetration depth of 1.5cm into flesh [2]. Cellular handsets have a power of 10mW to 2W peak using 450MHz to 2200Mhz and exhibit a penetration depth of 2.5cm in the middle of their range at 900MHz. So mobile cellular handsets give rise to a measurable heating effect of 0.1°C compared with no measurable increase for Bluetooth devices [2].

5 The Bluetooth Architecture

The Bluetooth Specification [3] defines all aspects of the Bluetooth technology including hardware and software components and shared processes and procedures.

Standard radio base-band, interface and software protocol specifications have been defined to enable a wide range of electronic devices to have Bluetooth functionality and to ensure interoperability between devices from different manufacturers.

The radio specification defines the frequency bands, channel arrangement and transceiver characteristics for a Bluetooth system.

The Bluetooth base-band specification defines packet formats, physical and logical channels and the different modes of operation that support the transfer of voice and data between devices.

Audio is needed to transfer voice. The specification for the Bluetooth voice interface follows both ITU-T and GSM recommendations.

5.1 The Bluetooth Device

In general terms a Bluetooth device is any complete electronic product that incorporates a Bluetooth radio. The radio itself is not the device it is only a component of the device.

In practical terms a Bluetooth device could be a mobile phone or a personal digital assistant (PDA) or a portable computer or even an IQ-Mover. As long as the complete product incorporates Bluetooth technology the item can be called a Bluetooth device. Naturally any Bluetooth device incorporates more technology than just the Bluetooth radio. Every device has its own distinct purpose a printer is supposed to print an IQ-Mover is supposed to move around and so on.

5.1.1 The Host and the Bluetooth Module

In Bluetooth terminology the non-Bluetooth part of a device is called the host.

5.1.2 The Module used at HTU

At HTU we have used the Bluetooth Application and Training Tool Kit from Ericsson. This is a hosted model (see chapter 5.1.3) the module only has the lower stack on its chip (see figure 5.3) and it only support point-to-point operation.

Figure 5.1 Bluetooth Application Tool Kit.

5.1.3 Three Different Bluetooth Implementation Models.

There are three implementation models [5] for the stack (the stack is described later in the thesis), dependent upon the functionality or resources the respective product has: hosted, embedded and fully embedded (see Figure 5.2).

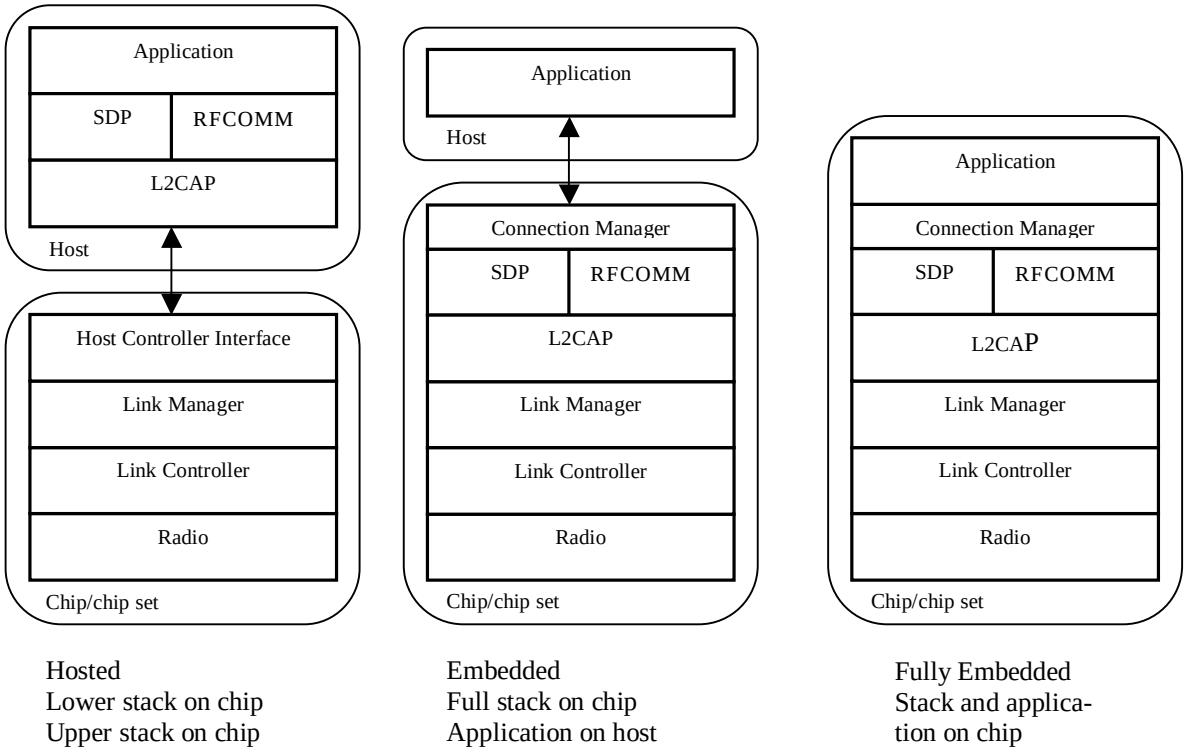


Figure 5.2 The three stack implementation models.

In the hosted model, the lower stack layers reside on the Bluetooth device, while the upper stack resides on a host (this may be a PC or a micro-controller if the product is mobile or standalone). They communicate via the Host Controller Interface, which sits between the lower layers and upper layers of the protocol stack forming a bridge between them. The two most common physical transports are UART (H4) and USB (H2).

The hosted model is optimum for applications where powerful host processors are already available and there is plenty of memory. Examples of hosted devices include USB dongles, PCMCIA cards, compact flash cards, V90 modems, Internet gateways and PC motherboards.

In the embedded model, the complete stack resides on a Bluetooth device but a separate user application is running on a host. This model is ideal for 2 and 3G mobile phones, ticket or vending machines, or PC peripherals that have limited processing power and available memory.

In the fully embedded model the complete stack and the user application are all on the Bluetooth device. There is limited memory resource on the Bluetooth device so any application will need to be relatively simple. The best example of a fully embedded device is

a headset. It has no need for complex processing, so the whole Bluetooth stack can run on the single microprocessor within the Bluetooth chip/chip set.

The lower stack up to the HCI is always provided with the Bluetooth chip/chip set, as it is unique to that silicon implementation. With the embedded model, the upper layers are also provided by the chip/chip set vendor – either free of charge if it is their own stack or there may be a license fee per device if they are using another vendor´s upper layers. The fully embedded model requires a silicon solution that allows the application code to be written and downloaded to it without compromising the integrity of the Bluetooth stack that should have already undergone the stringent qualification procedure. Any change to the stack requires it to be requalified!

The upper stack layers, above HCI, can be licensed from numerous vendors. Due to the inherent interoperability requirement of any qualified Bluetooth component, the choice is open. All of the available stack offerings “will” be conpatible with the chosen silicon´s lower layers. You can write your own upper layers but it will be a vast software undertaking - illustrated by the cost of licensing one. Protocol stack can be expensive but an expensive stack might just offer you extra features that help to sell your product [5].

5.2 The Lower Layers of the Stack

The lower stack controls all of the physical functionality, the radio, the base-band, and the Link Manager and Link Controller layers. Figure 5.2 shows the lower layers of one Bluetooth module.

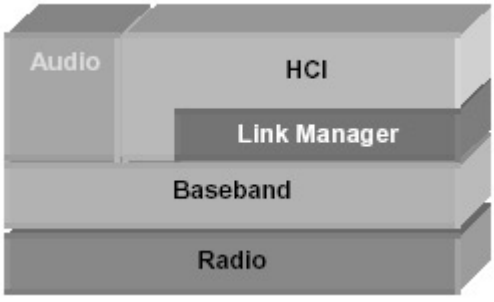


Figure 5.3 The different layers of the Bluetooth Module.

Bluetooth Training Tool Kit does not include the audio part.

5.2.1 The Bluetooth Radio

The Bluetooth radio is just like any broadcasting radio, except a whole lot smaller and less powerful. A Bluetooth radio is small enough to fit between the gap in your two front teeth and broadcasts a minuscule 1 milliwatt. The Bluetooth radio transmits over the 2.4GHz RF band, utilizing spread spectrum frequency hopping and Time Division Duplexing (TDD) technologies. A Class 2 Bluetooth radio (the most common type of device) can function over a 10-meter range.

There are actually three different classes of Bluetooth radios defined in the Bluetooth Specification (see table 5.1). Power output and thus send/receive range is the differentiator between the classes.

Table 5.1 Output Power for Each Bluetooth Power Class.:

Device Power Class	**Max Output Power (mW)**	**Max Output Power (dBm)**	**Expected Range (*?Obstructed Environment?*)**
Class 1	100mW	20dBm	100m
Class 2	2.5mW	4dBm	10m
Class 3	1mW	0dBm	10cm

dBm (decibel relative to one milliwatt) is an alternate measurement of output power.

Unlike the radio in your audio system the Bluetooth radio functions both as a transmitter and a receiver.

5.2.2 Bluetooth Baseband

The Baseband is the physical layer of the Bluetooth. It manages physical channels and links apart from other services like error correction, data whitening, hop selection and Bluetooth security. The Baseband layer lies on top of the Bluetooth radio layer in the Bluetooth stack. It is implemented as a Link Controller, which works with the link manager for carrying out link level routines like link connection and power control. The Baseband also manages asynchronous and synchronous links, handles packets and does paging and inquiry to access and inquire Bluetooth devices in the area. The Baseband transceiver applies a time-division duplex (TDD) scheme, (alternate transmit and receive). Therefore apart from different hopping frequency (frequency division), the time is also slotted (see chapter 4.6).

5.2.3 The Link Manager (LM)

It is the LM that translates HCI (see 6.1.1) commands into operations at the baseband level, managing the following operations.

- Attaching Slaves to a piconet.
- Breaking connections to detach Slaves from a piconet.
- Configuring the link including controlling Master/Slave switches (where both devices must simultaneously change roles).
- Establishing ACL and SCO link.
- Putting connections into low-power modes: Hold, Sniff and Park. (I don't write about these modes because the Bluetooth Module I used does not support them.)
- Controlling test modes.

5.2.4 The Host Controller (HC)

The host Controller (HC) (in figure 5.3 named HCI) is the part of the Bluetooth that manages all communication and interaction between the Bluetooth module and the host device. The connection between the two parts can be hardwired (that is the Bluetooth module can be built into the host device's main circuit board) or modular (with the Bluetooth module attached to the host device as an add-on accessory or plug-in card).

5.2.4.1 Bluetooth Traffic Light Utility (BTU)

It is important to know that all Ericssons released Modules do not fit whit all versions of their Stack. I had some trouble before I noticed the Tool Bluetooth Traffic Light Utility.

Figure 5.4 BTU main window.

This program is used to determine the compatibility between the used “PC Reference Stack” and the connected Module. The version number for “PC Reference Stack” (in the application called “Host Stack”) is picked up from the stack. From the Module the following information is retrieved:

- Bluetooth address
- Manufacturer number
- HCI-version number
- LMP-version number
- HCI-revision number
- LMP-subversion number

You can read more about this in User Manual [6].

5.3 The Bluetooth Profiles

The purpose of a profile is to provide a clear description of how a full specification of a standard system should be used to implement a given end-user function. If everyone deploys a particular communications standard in the same way, then each product so created should be able to interoperate. This notion of profiles originated from the International Organisation for Standardisation (ISO/IEC TR10000).

Bluetooth profiles work in just the same way. They ensure interoperability by providing a well-defined set of higher layer procedures and uniform ways of using the lower layers of Bluetooth. By doing this the Bluetooth profiles provide a way for Bluetooth

technology to slot into different devices and applications, yet still work in standardised ways. For example a Bluetooth headset purchased form Manufacturer A will interwork with a Bluetooth enabled cellular phone purchased from Manufacturer B.

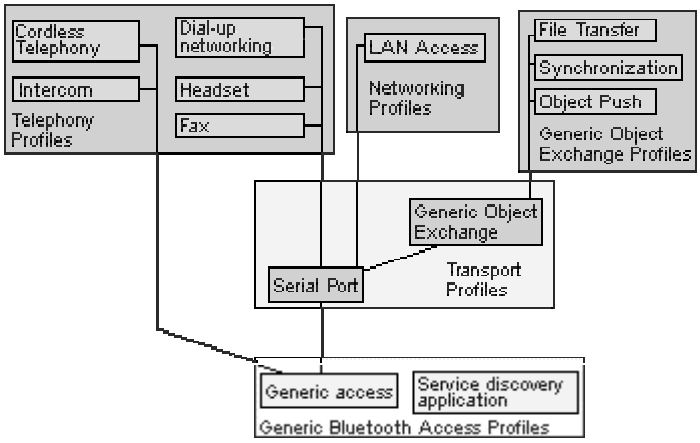


Figure 5.5 Bluetooth profiles implemented by Bluetooth version 1.0.

Figure 5.5 shows how Bluetooth profiles are organised into groups, with each profile building upon the one beneath. For developers this means that key features of one Bluetooth solution can be recycled in other solutions bringing down development costs and speeding up the development cycle.

The Generic Access Profile (GAP) is the most basic Bluetooth profile all other profiles are built upon it and use its facilities.

5.3.1 New Profiles

A series of SIG working groups continue the process of producing and refining the Bluetooth specification. They will come with new profiles ahead.

6 The Ericsson System Level Description

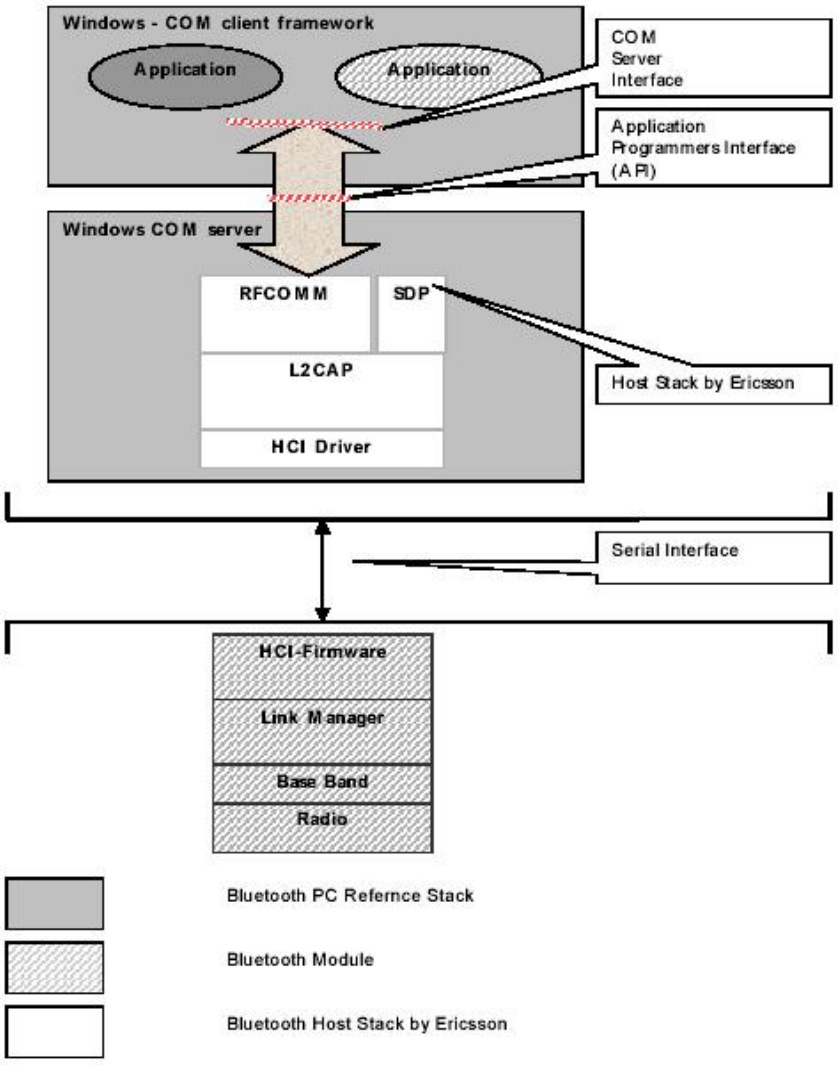


Figure 6.1 An overview of the PC Reference Stack and its components.

6.1 The Upper Stack

The upper stack that is delivered together with the Ericsson Bluetooth Module is the Bluetooth PC Reference Stack by Ericsson.

The Stack gives the costumer a ready to use module containing selected higher-level Bluetooth protocols to be integrated in a Windows environment. The Bluetooth PC Reference Stack will run in a Windows NT, 98 or 2000 environment.

The Bluetooth PC Reference Stack by Ericsson (the host stack in figure 6.1) containing the following higher layers/Interfaces of the Bluetooth protocol stack compliant to the Bluetooth foundation specification:

- RFCOMM (is a transport protocol)
- SDP (Service Discovery Protocol)
- L2CAP (Logical Link Control and Adaptation Protocol)
- HCI Driver (Host Controller Interface)
- OBEX (OBject EXchange protocol)
- TCS (Telephony Control protocol Specification)

6.1.1 HCI (The Host Controller Interface)

Named HCI Driver in Bluetooth PC Reference Stack by Ericsson.

To insure interoperability of Bluetooth modules from various manufacturers the Bluetooth Specification defines a standard interface (and Communications protocol) that can be used by all Bluetooth modules and by all host devices that incorporate the Bluetooth technology.

The Bluetooth standard for the HCI defines the following:

- Command packets used by the host to control the module.
- Event packets used by the module to inform the host of changes in the lower layers.
- Data packets to pass voice and data between host and module.
- Transport layers, which can carry HCI packet flow.

This Host Controller Interface (HCI), while not a mandatory component of the Bluetooth Specification (it is not completely necessary if the Bluetooth module is fully integrated into the design of the host device), is useful for those Bluetooth devices that are meant to be used in an add-on or accessory fashion.

6.1.2 L2CAP (Logical Link control and Adaptation Protocol).

L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassemble operation, and group abstractions. L2CAP permits higher-level protocols and applications to

transmit and receive L2CAP packets up to 64 Kilobytes in length. L2CAP only supports ACL links.

All applications must use L2CAP to send data. Bluetooth´s higher layers such as RFCOMM and SDP also use it, so L2CAP is a compulsory part of any Bluetooth system.

The idea behind L2CAP is to provide an interface similar to TCP/IP function calls.

6.1.3 RFCOMM

RFCOMM is a transport protocol, with additional provisions for emulating RS-232 serial ports. The RFCOMM-protocol support up to 60 simultaneous connections between two BT devices. For the purposes of RFCOMM, a complete communication path involves two applications running on different devices with a communication segment between them. RFCOMM is able to transmit/receive data packets of up to 32 KB over such a link. In this context, the term application may mean other things than end-user application, e.g. higher layer protocols or other services acting on behalf of end-user applications.

6.1.4 SDP (Service Discovery Protocol)

Discovery of services is a crucial part of the Bluetooth framework. Service discovery is fundamental for all the usage models. Bluetooth is designed to allow you to walk into an area and find a printer, without having to preconfigure settings. When you used the printer, you can walk away and forget its details. SDP is the part of Bluetooth that allows this to happen. The SDP provides a means for applications to discover which services are available and to determine the characteristics of those available services using an existing L2CAP connection. After that, an appropriate separate connection between two or more Bluetooth devices can be established using information obtained via SDP. The service discovery application does not make use of SDP as a means of accessing a service, but rather as a means of informing the user of a Local Device the services that are available to his/her device by (and possibly via) Remote Device(s).

6.1.5 OBEX

OBEX is based on the IrOBEX protocol, which is a compact, efficient, binary protocol that enables a wide range of devices to exchange data in a simple and spontaneous manner. Spontaneity is

important for Bluetooth, where short duration ad-hoc connections can be set up.

OBEX runs on top of RFCOMM.

6.1.6 TCS (Telephony Control protocol Specification)

TCS is using point-to-point or point-to-multipoint signalling to establish speech and data calls between Bluetooth-devices. The calls are mapped towards the L2CAP-layer, point-to-point via a connection-oriented channel, point-to-multipoint via a connectionless channel, sent as a piconet broadcast.

6.2 COM-server

The Ericsson Bluetooth Pc Reference Stack is a COM-server, containing the Bluetooth HOST Stack. Because of that your application has to be a COM-client to be able to communicate with the HOST Stack.

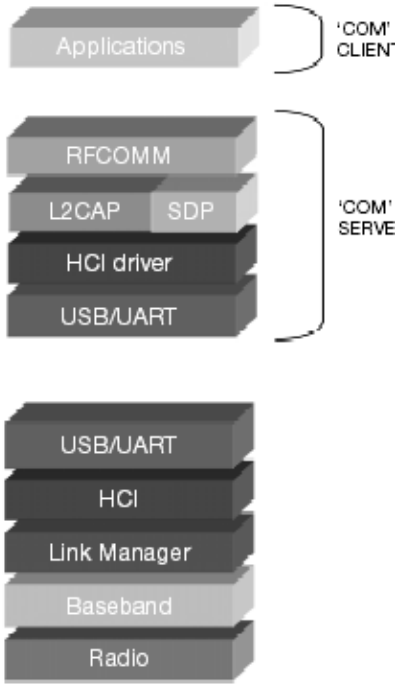


Figure 6.2 The Com-server and Com-client on top of the Module.

A way to simplify this is to make calls directly to the HOST Stack APIs. To do that a couple of files have to be included in the main-file in your application. These files are:

- 'bt_interface.lib',
- 'serverevents.cpp',
- 'serverevents.h' and
- header files located in the exp-folder in the delivery structure.

The file 'serverevents.cpp' creates a COM-client shell to enable communication with the HOST Stack. For example, see the two application suites ('TestSample' and 'Chat') containing the delivery of the Bluetooth Pc Reference Stack by Ericsson. They are implemented this way. You can read more about this in User Manual [6].

6.3 HCI packets

There are three different types of HCI packets. HCI commands packets are sent from the host to Bluetooth module and HCI event packets, which are sent from the Bluetooth module to the HCI host. Finally there are HCI data packets.

6.3.1 HCI Command packet

This is used for sending commands to host controller from the host. The format is shown in Figure 6.3. Each command consists of 2 bytes Opcode used to identify different types of commands. The Opcode parameters are divided into two fields: Opcode Commands Field (OCF) and Opcode Group Field (OGF).

The OGF occupies the upper 6 bits of the Opcode and the OCF occupies the other 10 bits of the Opcode. Further more all of the command parameters for each command have fixed size. Each command will specify each parameter and size of each parameter contains the command.

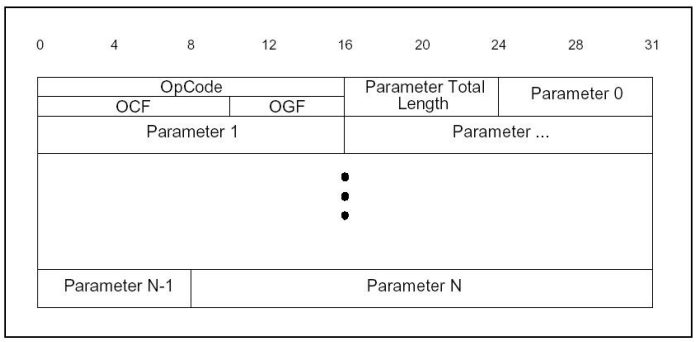


Figure 6.3 HCI command packets.

6.3.2 HCI Event packets

Used by the host controller to notify the host when events occur. The format of the HCI Event packet is shown in figure 6.4. The Event occupies 1 byte of the packet and the parameter length 1 byte.

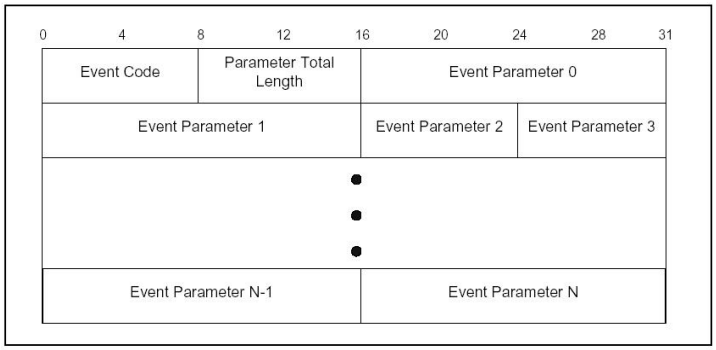


Figure 6.4 HCI Event packet.

6.3.3 HCI Data packet

Used for exchange data between the host and host controller. There are two types defined depending on the connection. The format for the ACL is shown in Figure 6.5 below. It consist of 14 bits for the connection handler, 2 bits for the flag, the total length of the data that is in the packet and the data it self.

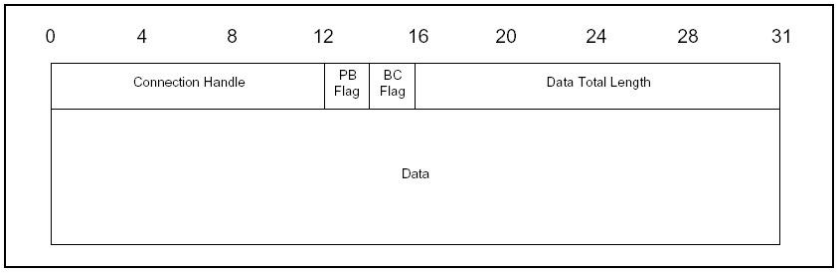


Figure 6.5 HCI ACL Data packet.

HCI Data Packets are used to exchange data between the Host and Host Controller. The data packets are defined for both ACL and SCO data types.

The Flag Bits consist of the Packet_Boundary_Flag (PB) and Broadcast_Flag (BC). The Packet_Boundary_Flag is telling which number the packet is for example if it is the first, the last or someone in the middle. The Broadcast_Flag says if the packet is going to be broadcast sent or point-to-point sent.

The SCO Data Packet is shown in Figure 6.6.

Figure 6.6 HCI SCO Data Packet.

The Reserved Bits consist of four bits they are reserved for future use. Length of SCO data measured in bytes

6.4 API (Application Programmers Interface)

With the Tool Kit comes a special interface to the stack it is named the BTAPI. The interface is an Ericsson propriety API, designed as environment independent and flexible as possible [6]. It allows the establishment and tear down of data and voice connections, sending and receiving of data, as well as other Bluetooth functions such as performing inquiries and page function. Dynamically loading profile and adding own API complements suitable for the selected environment. One other important feature is the ability to access the L2CAP layer directly in order to implement new protocols on top of L2CAP by the user.

6.4.1 API supported profiles

- Dail-up networking
- Fax
- Headset
- LAN Access
- Serial Port

Missing profiles can be added by copying an existing profile routine and adapt it to your wishes. You can read more about this in the Users Manual [6].

6.4.2 The stack API

The stack API gives application programmers access to the APIs provided by the different layers of the stack; RFCOMM, SDP, L2CAP, SCM and HCI interfaces. By using the API the application programmer is given full access to the features provided by the Bluetooth technology. In the description that follows when

discussing different layer such as layer A and layer B etc, the layers can be any of the components of the stack.

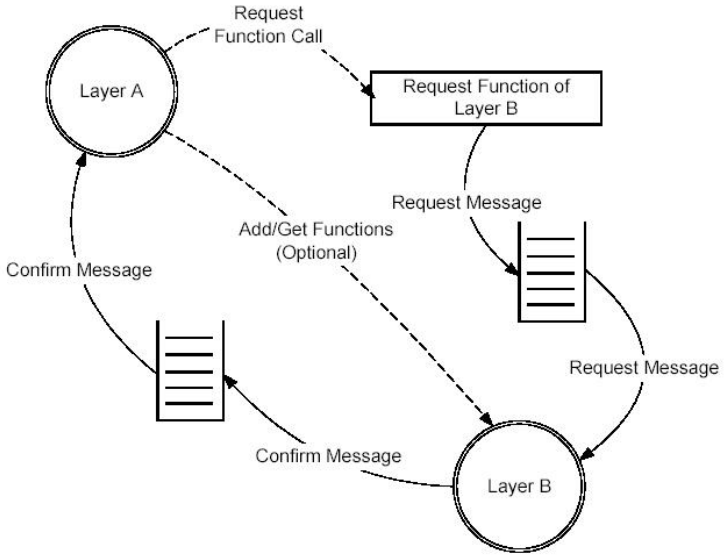


Figure 6.7 Inter-process communication within the Bluetooth Host Stack.

Layer A communicates with layer B by calling the correct request message function, which is the same for every layer. The request function allocates memory to send data and parameters, which in the next step is sent to layer B. Layer B waits for messages from the queue, when a new message arrives to layer B it's handled in a state machine. Depending on if the message results in a confirmation, a message is send directly to layer A. In case it results in another message to another layer, then layer B calls the correct request function. All the message handling is done by the operating system routines.

6.4.3 Global API Description

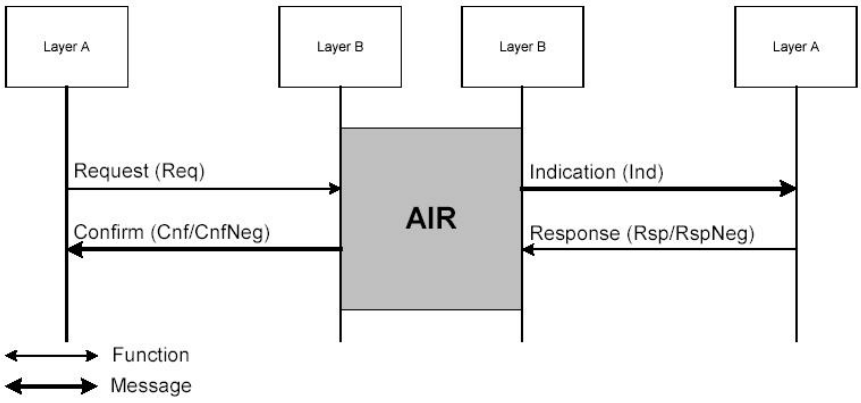


Figure 6.8 Communication principles between different layers.

Control message is the communication between different layers, which exchanges information with each other. A data message passes through the stack where each layer adds its header to the data packet. In the same way, on the other side each layer removes the headers one by one. The difference between command messages and event messages can be seen below. A command message is messages sent from a higher layer to a lower layer, which is a request. An event message is the opposite, a lower layer sent a message to a higher, corresponds to an indication.

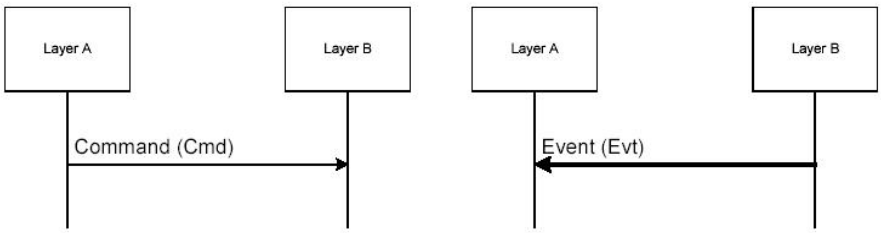


Figure 6.9 Command & Event.

6.4.4 Layer to layer

Communication between two protocol layers is described here. Because all the protocol components have the same kind of messages one could use the standard message to communicate with all the other different protocols, in another word replacing the "B_" with SCM_, COM_, SD_, L2CAP or HCI_. Table 6.1 shows the messages, which can be sent from layer A to B.

Table 6.1 Communication from layer A to layer B.

Name	Function Description
B_ReqStart	Start message to start component
B_ReqRegister	Control function for set up control connection between two layers
B_ReqDeRegister	Control function for control disconnection between two layers
B_ReqConnect	Control function for set up connection with task B (peer to peer)
B_RspConnect	Control function for set up connection successful with peer
B_ReqDisconnect	Control function for disconnection with task B (peer to peer)
B_RspDisconnect	Control function for disconnection successful with task B
B_DataSend	Data function for sending data to the other peer

B_RspData	Data function for successfully sending data to peer
B_DataAlloc	Allocates a buffer containing a MSG_TDataMsg and a Length in bytes
B_DataFree	Frees buffer created with B_DataAlloc. Pointer is a byte array to the data buffer
B_DataExtract	Extract the header information and data from ptDataInd. Function returns length and pointer to the start of the data

Table 6.2 Communication from layer B to layer A.

Name	**Message Description**
B_START_CNF	Start message confirm for start component
B_REGISTER_CNF	Control message for confirm set up control connection
B_REGISTER_CNF_NEG	Control message for negative confirm set up control connection
B_DEREGISTER_CNF	Control message for confirm control disconnection
B_DEREGISTER_CNF_NEG	Control message for negative confirm control disconnection
B_CONNECT_CNF	Control message for confirm set up connection with task B
B_CONNECT_CNF_NEG	Control message for negative confirm set up connection with task B
B_CONNECT_IND	Control message for set up connection at peer side
B_DISCONNECT_CNF	Control message for confirm disconnection with task B
B_DISCONNECT_CNF_NEG	Control message for negative confirm disconnection with task B
B_DISCONNECT_IND	Control message for disconnection at peer side
B_DATA_CNF	Data message for confirm sending data to the other peer
B_DATA_CNF_NEG	Data message for negative confirm sending data to the other peer
Name	**Message Description**
B_DATA_IND	Data message for sending data at peer side

6.4.5 Start

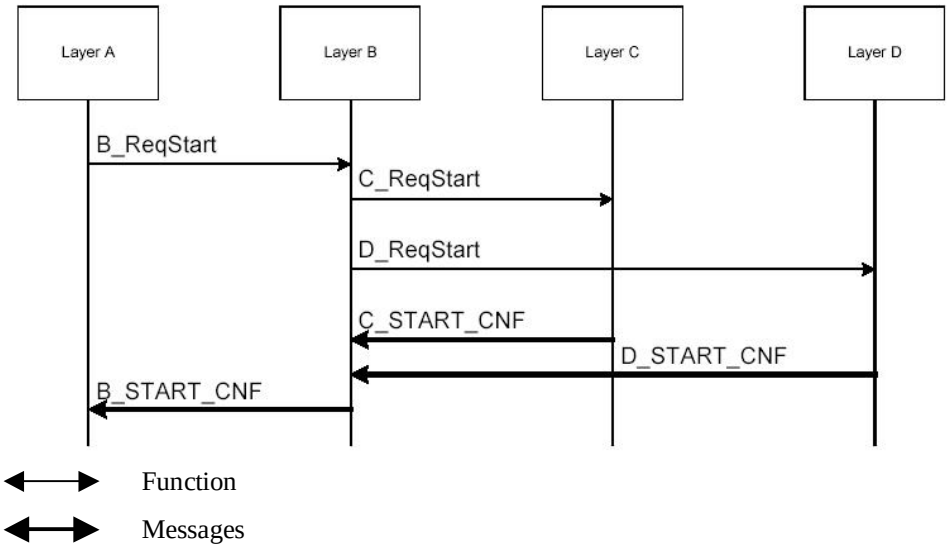
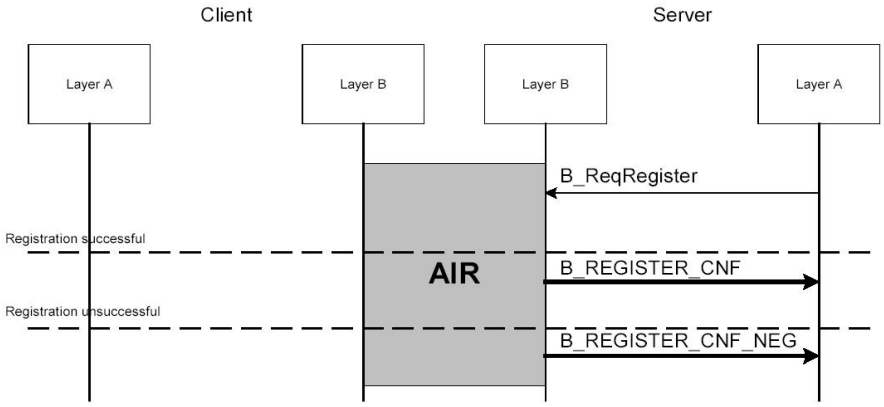


Figure 6.10 The start procedure.

Before making any use a component one has to start the interface to the lower layer, layer A starts the interface with layer B etc. This is done by function B_ReqStart. Figure 6.10 shows the start procedure.

6.4.6 Registration & DeRegistration



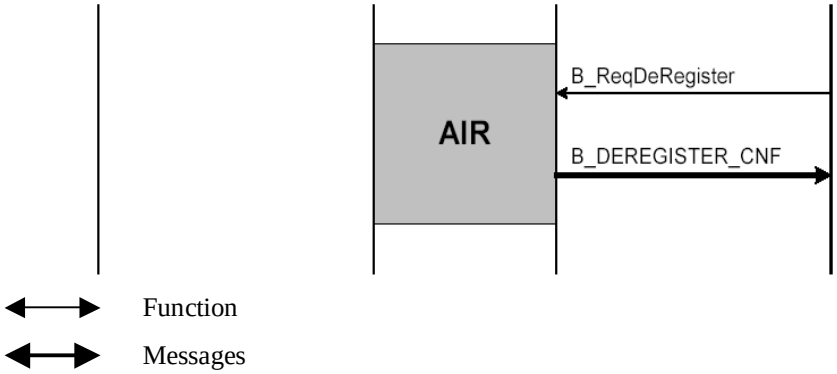


Figure 6.11 Registration and DeRegistration.

The registration is next step; a server must register to lower level with B_ReqRegister. As the lower level must know were to send the received messages. A Client layer does not need to register itself, because it receives messages from higher layer and thereby knows were to send back the response. If the registration is successful the two layers can set up a connection and send data. One important note, sever registration has no effect on the client side. A de-registration is done by sever sending B_ReqDeRegister. See Figure 6.11.

6.4.7 Connection & DisConnection Set up

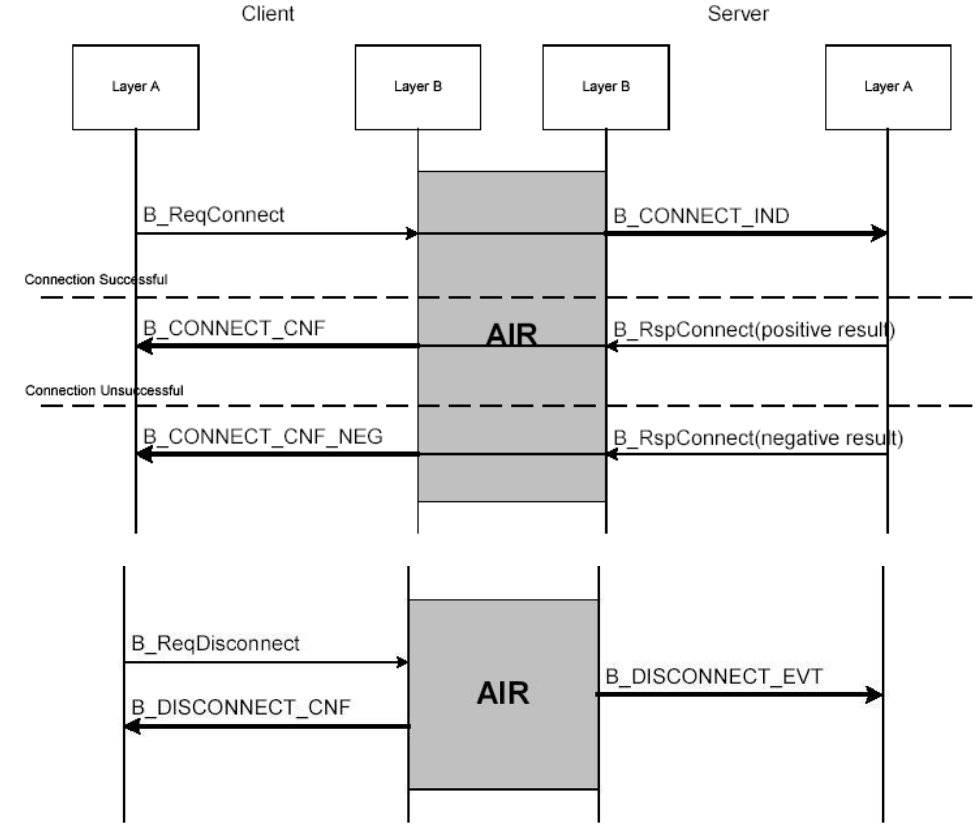


Figure 6.12 Connection and disconnection sequence.

A client initiates the set up connection by calling the B_ReqConnect function of the protocol layer it wants to use. If the sever is registered to the lower layer a B_CONNECT_IND is received. As seen above the server can decide whether to accept or to decline, depending what the server send as B_RspConnect parameters. Either client or the server can send a B_ReqDisconnect; the other side receives a B_DISCONNECT_EVT.

6.4.8 Data transmission

When the connection is established, both the server and the client can start sending data to the other side. The highest layer must call the B_DataAlloc function, which allocates memory in the stack and the highest layer can put its data in there. After this a B_DataSend is send to a lower layer. This function call has a pointer to the allocated memory as parameter. This means that the data is not copied on every level as the data travels though the layers. On the opposite side a B_DATA_IND is received, which indicates that data is delivered to layer A. This layer then sends a B_DataExtract to retrieve a pointer to the data block and the length of it. If this is successful then a B_RspData is sent. Depending on the correctness of the data a positive or negative response is indicated I the return parameter. Figure 6.13 shows a data-sending scheme.

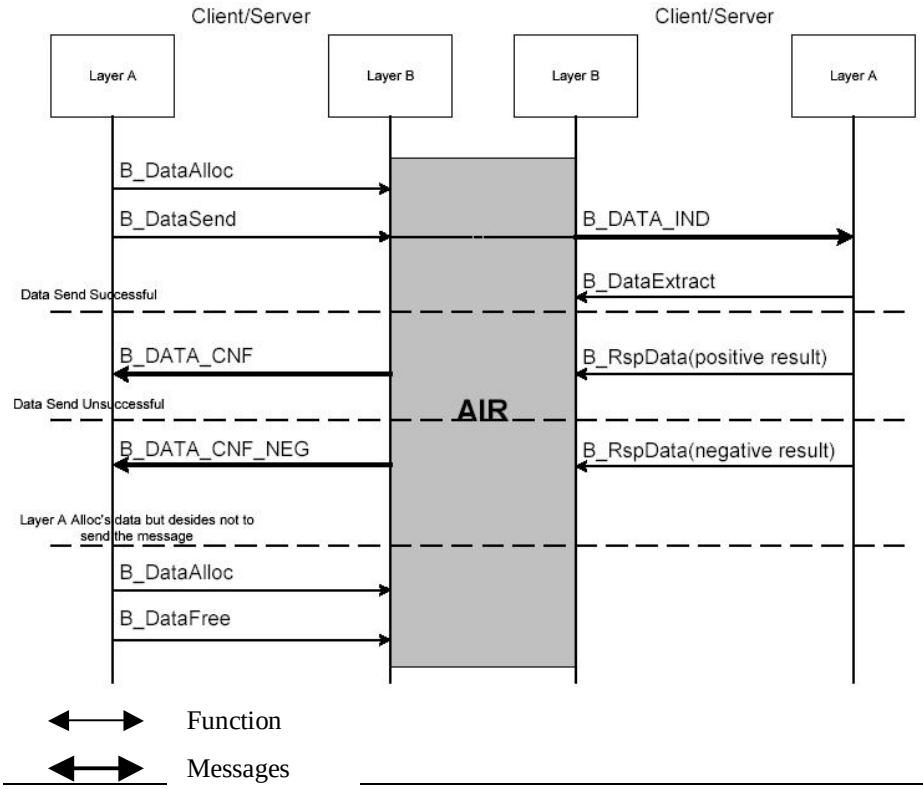


Figure 6.13 Data transmission.

6.4.9 The Page and Inquiry Procedures

To establish new connections between Bluetooth devices in order to form a piconet, the page and inquiry procedures are used.

Page is used to contact a Bluetooth unit you already know the BD_ADDR and inquiry is to request the BD_ADDR of units in the area.

6.4.9.1 The Inquiry Procedure

The units in range can respond to an inquiry. The figure 6.14 shows the Inquiry procedure and answer from two different devices. The units respond with their BD_ADDR and the CLK so the inquiry unit can have knowledge on the address and the clock of those units that responds to the inquiry.

The ucInquiryLength specifies the total duration of the Inquiry. ucNrofResponses specifies the number of responses that must be received before the inquiry is halted. An ucNrofResponses of zero indicates unlimited responses.

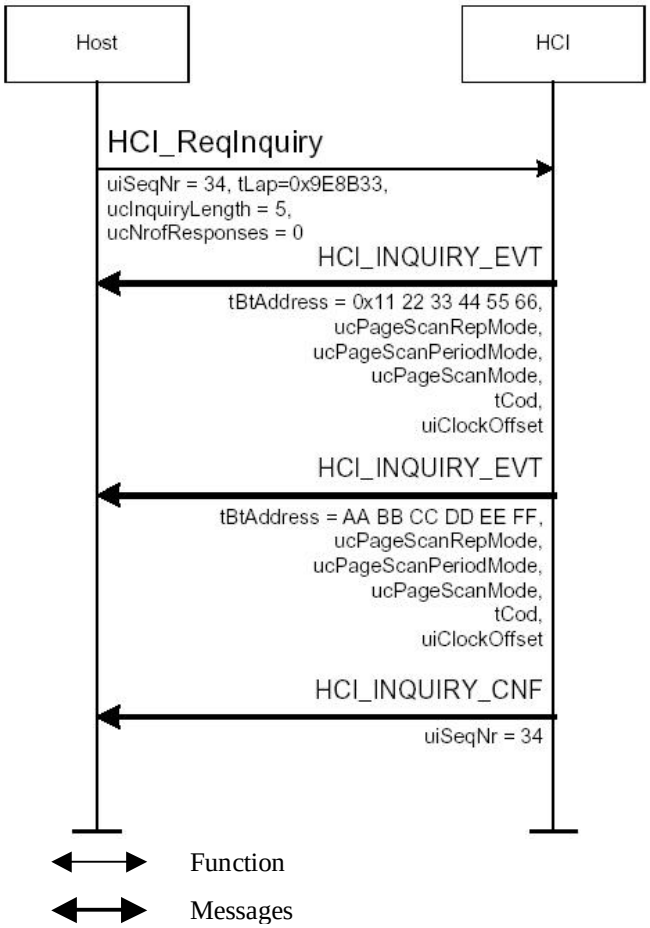


Figure 6.14 Inquiry.

6.4.9.1.1 Inquiry Result Filter

An Inquiring device may not be interested in every device in the area. To provide the host with some control over the result it sees, the HCI_Set_Event_Filter command can be used to set an inquiry result filter, which means that some devices will not be reported in HCI_Inquiry_Result_Events. Inquiry results can be filtered out using three criteria:

- Report only new devices not seen before.
- Report only devices with a specific device type.
- Report only the device with a specified Bluetooth device address.

Of course this filtering could be done at the Application Level instead of the HCI Level.

6.4.9.2 The Page Procedure

It is possible to carry out the paging procedure with no prior inquiry if the device address is already known, as may be the case with the IQ-Mover. The paging device is able to address the intended Slave, but because no CLKE (estimate of the Slave´s clock) is available, the procedure will take much longer. In theory, the maximum length of time is 10s.

After that the connection can be established (by then using the Page command).

When paging is completed, a connection between the two Bluetooth devices can be established.

This includes:

- link establishment: The physical link is a sequence of transmission slots on a physical channel alternating between master and slave time slots.
- channel establishment: A logical link is established to setup a Bluetooth channel between the two devices.
- connection establishment: Finally, a connection between the applications on the two Bluetooth devices is established.

In the connection state, a connection has been established and packets can be sent back and forth between the master and slave devices.

6.4.9.3 Avoiding collisions

In order to avoid collisions between devices that wake up in the same inquiry hop channel, a protocol is used which ensures that each time a slave responds to the same inquiry message it does so on different frequencies and at different times.

7 TEST SAMPLE APPLICATION SUITE

In Ericssons Bluetooth Tool Kit two applications are included.

- The TestSample Application Suite

- The Chat Application Suite

The Test Sample Application is intended as guidance when designing own applications for non-windows environment making use of Ericssons PC reference stack as a development tool. When writing windows specific programs it is better to take the Chat application as guidance.

I have modified the TestSample Application in order to get applications (Master and Server applications) for our IQ-Mover. (See appendix). The Security Application did I use right as it was.

7.1 Test Sample application

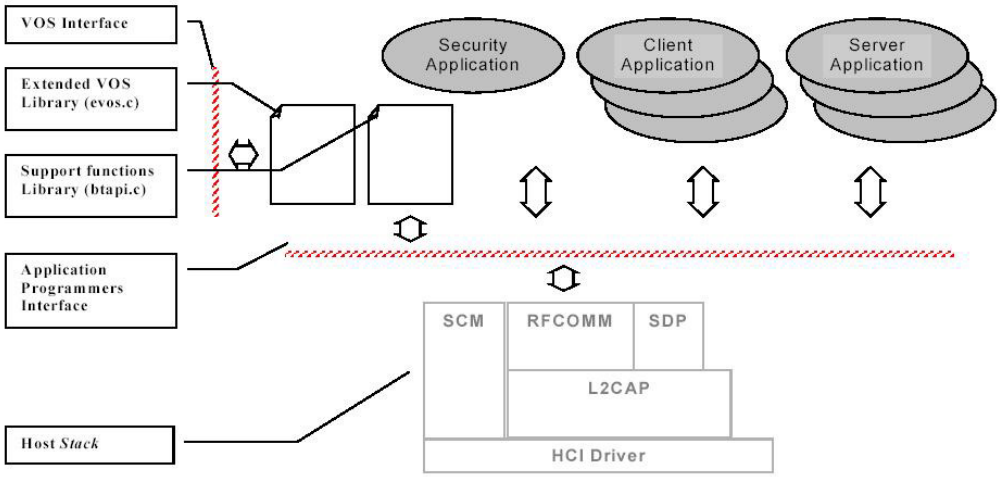


Figure 7.1 How the different parts of the sample application interacts with each other, the stack and VOS.

The Test Sample Application consists of three parts, the first application acting like a server (slave), the second application acting like a client (master) and the third one handling the BT security.

Table 7.1 File description of the Test Sample Application.

Sample Component	File name	Description
Client side application	client.c, client.h	This module contains an example of a client type of application, running on top of RFCOMM. This client expects a headset server in a fixed destination device.
Server side application	server.c, server.h	This server acts as a headset server.
Security module.	security.c, security.h	This component takes care of accepting a new ACL connection and provides the stack with a PIN-code when necessary. The stack supports several client and server sessions at a time but only one can be the security manager. The security manager in this sample is a stand-alone application.

You can read more about the original Test Sample Application in the User Manuals [4] and [6].

7.2 My Modified Test Sample

I have used the inquiry, page and connection functions as they were. I have also used the headset profile and SCO link as they were. The echo function did I take away because we don not want handshakes. I have placed a timer just after the SendData function and investigate a new event (TIMEOUT_2_EVT). This makes a loop of data transmissions. An A is transmitted every 1/10 second. This ends when anyone turns the disconnect button. The time between the transmissions is easy to change.

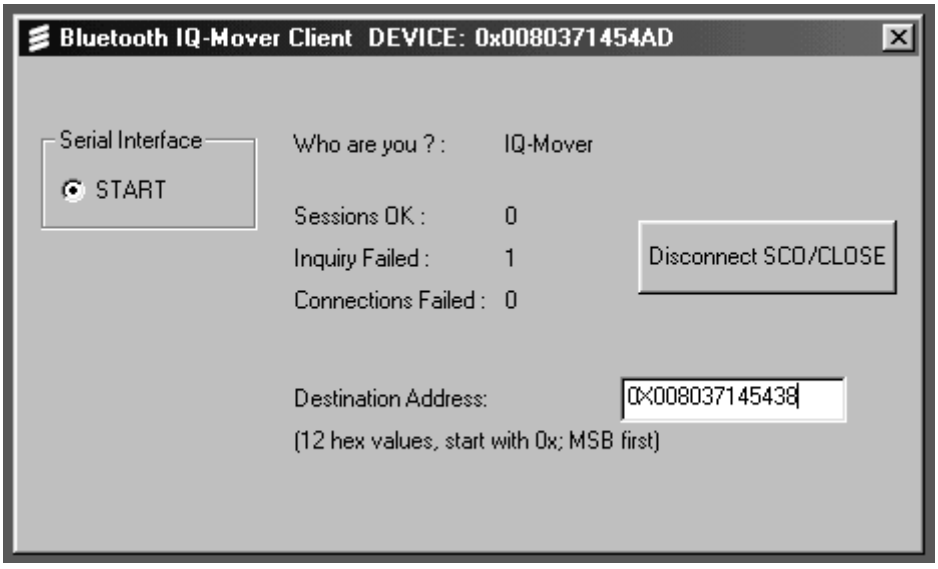


Figure 7.2 My modified Test Sample Client.

Figure 7.3 My modified Test Sample Server.

The module has to be connected to the PC with the serial port after the modification.

8 Bluetooth and FPGAs

For a technology such as Bluetooth, which is not terrible demanding of current ASIC technology, a good FPGA implementation is very feasible [2].

Most Bluetooth IP suppliers at present have demonstration systems that utilise high performance FPGAs, and these are able to promote confidence in the IP, prove the technology, and provide better coverage of bugs and system testing. Of course, there are some “porting” issues to be addressed when moving to an ASIC implementation such as embedded ASIC RAM block. It is particularly important to ensure that enough validation is carried out on the ASIC design itself and not just to rely on the lab bench verification of the FPGA based system.

9 Results

I have managed to modify the Test Sample Application in order to send one sequence every second to one beacon. An SCO link was used in order to get rid of retransmissions. The Bluetooth Modules I used were only point-to-point so for that reason I have not been able to try to connect to more then one beacon. The Ericsson Module has only the lower layers the upper layer has been placed in a PC.

10 Discusion

Bluetooth is still a new technology. It will take some time before this will be as easy to use as the SIG would like it to be. There will be new functions added for a long time I think.

The optimal solution for the IQ-Mover would be an ACL link witch does not retransmit all packets. The AUX1 is that kind of packet but it does not have any type of bit-error control in the transmitted data sequences. I have not been able to try the AUX1 packet type. There is an inconsistency in the Ericsson stack implementation. I have emailed them about this and they say they are going to correct it in the future.

If the AUX1 packet is not Ok you can try to use SCL links instead. If SCO links are used not more then three beacons can be connected at a time. The IQ-Mover then have to stop when it is time to change from one beacon to another. This is because it will take some second (up to 20s [5]) to find and connect to a new one. This stop will probably be shorter if AUX1 packets can be used.

If FPGAs are going to be used the embedded architecture will be the best I think. This can be something for a student who knows more about FPGAs than I do to study in a coming project. HTU then have to by a Bluetooth module for embedded solutions. Bluetooth Development Kit from Teleca is a module for embedded solutions; the price is today 6500 USD.

Another thing that would be interesting to look at are the different profiles. Can the Generic Access Profile be something for the IQ-Mover and of course it would be important to test the use of a multipoint device in order to connect more then one beacon to the IQ-Mover. The connection to the second beacon will be done in the same way as the first they answered me when I mailed the support regarding this. You only have to change the BD-Address.

To understand Bluetooth I have been forced to learn more about C++, especially classes and pointers. I have not used Visual C++ before and that was a big step to understand the program before I could study the Bluetooth Test Sample application in order to modify it.

11 Conclusions

It will be possibly to use Bluetooth in the IQ-Mover system.

12 References

[1] Miller Michael. Discovering Bluetooth. USA 0-7821-2972-2

[2] Bray J, Sturman C. Bluetooth connect Without Cables. USA 2001.

[3] Bluetooth Special Interest Group, "Specification of the Bluetooth System version 1.1". Bluetooth Special Interest Group 22 February 2001.
Available from www.bluetooth.com

[4] Users Manual –Ericsson Bluetooth PC Reference Stack R1C, Ericsson Mobile Communication AB, 19 January 2001.

[5] Bray J. Bluetooth Application Developer´s Guide: The Short Range Interconnect Solution. USA 2002.

[6] Users Manual –Ericsson Bluetooth PC Reference Stack R2B, Ericsson Mobile Communication AB, 14 May 2001.

13 Appendix

1. MSC IQ-Mower
2. BT_TestSampleClientDlg.cpp : implementation file
3. client.c
4. BT_TestSampleServerDlg.cpp : implementation file
5. server.c